Towards Data Quality Runtime Verification

Janis Bicevskis
Faculty of Computing
University of Latvia, Latvia
Email:
Janis.Bicevskis@lu.lv
ORCID: 0000-0001-5298-9859

Zane Bicevska
DIVI Grupa Ltd,
Latvia
Email:
Zane.Bicevska@di.lv
ORCID: 0000-0002-5252-7336

Anastasija Nikiforova
Faculty of Computing
University of Latvia, Latvia
Email:
Anastasija.Nikiforova@lu.lv
ORCID: 0000-0002-0532-3488

Ivo Oditis
DIVI Grupa Ltd,
Latvia
Email:
Ivo.Oditis@di.lv
ORCID: 0000-0003-2354-3780

Abstract— **This paper discusses data quality checking during business process execution by using runtime verification. While runtime verification verifies the correctness of business process execution, data quality checks assure that particular process did not negatively impact the stored data. Both, runtime verification and data quality checks run in parallel with the base processes affecting them insignificantly. The proposed idea allows verifying (a) if the process was ended correctly as well as (b) whether the results of the correct process did not negatively impact the stored data in result of its modification caused by the specific process. The desired result will be achieved by use of domain specific languages that would describe runtime verification and data quality checks at every stage of business process execution.**

Keywords—data quality, runtime verification, business process, domain specific languages

I. Introduction

NOWADAYS, the most part of processes is based on more than one information system or service. Moreover, the environment where processes are running usually is very heterogeneous. As a result, besides users, other information systems and changes made in them may affect execution of the initial process. One of the typical solutions for such situations is detection of incorrect execution by system monitoring or support staff, however, identification of affected business processes isn't possible in this case. As a result, the necessity for runtime verification of business processes appears to keep the process consistent at any time [1]. As it was discussed in [2], runtime verification of business processes allows (a) detection of incorrect execution that is possible in the case of system monitoring, but also (b) identification of business processes that may be affected by it. As a result, asynchronous runtime verification of business process was proposed focusing on timely and accurate problem identification.

However, runtime verification of business processes checks only the correctness of process execution based on the evaluation of process execution sequence, meanwhile this research proposes to extend it by involving data quality mechanism, that will assure the specific process did not negatively affect data stored in information systems that were affected by this process. It is achieved by applying data object-driven approach to data quality evaluation [3]. This approach is based on definition of data object which quality should be analysed, quality requirements definition that are applied to the parameters of the defined data object and measuring data quality. In scope of the proposed solution, data object is derived from data that were affected by running process. Data quality requirements are defined to check whether data are still correct and "external constraints" are still valid.

The paper deals with following issues: concepts used for data quality checks in the runtime verification (Section 2), idea and main concepts of the proposed solution (Section 3), analysis of the proposed solution (Section 4), conclusions and future work (Section 5).

II. Basic Concepts

This chapter briefly discusses the concepts used in runtime verification and data quality research that are necessary for discussing the ideas and solutions proposed.

A. Runtime Verification

Runtime verification mechanism proposed in [2], doesn't intervene into execution of processes. It observes processes from the aside, collecting and verifying events confirming process step execution, in accordance with business process description. The main point is checking of the verification of business process execution in compliance with the process verification description. The description of the verification process must specify two aspects: (a) event confirming step completion, and (b) the time when each step in the process must be finished. Two main components of this mechanism are agents and controller.

The agent plays a role of event detector. It is software that checks the occurrence of a specific event. An example of such event can be record insertion. All events detected by

The research leading to these results has received funding from the research project "Competence Centre of Information and Communication Technologies" of EU Structural funds, contract No. 1.2.1.1/18/A/003 signed between IT Competence Centre and Central Finance and Contracting Agency, Research No. 1.7 "The use of business process models for full functional testing of information systems".

event agents are sent to the centralized controller for verification.

The controller analyses process verification descriptions, collecting event messages that were sent by agents and verifies flow compliance with the verification description.

Agents are developed for different components (databases, file systems, email servers etc.) and not implemented into software under verification. Thereby proposed mechanism allows verifying business processes executed by more than one system, running over several platforms, and even provided by more than one operator.

Runtime verification of business processes allows detection of incorrect execution and identification of business processes that may be affected by it. Both, detection of incorrect execution and identification of business processes that may be affected by it, take place immediately after changes were made by including appropriate checks in the runtime verification of business processes. In comparison with more traditional for such cases system monitoring, it allows to fix the occurrence of such problem as soon as possible for its timely solving to achieve as high result as possible. Moreover, identification of business processes that may be affected by it usually isn't considered at all. In other words, asynchronous runtime verification of business process focuses on timely and accurate problem identification.

Significant benefit of this approach is that it can be used when existing software does not respect any component addition. It is very useful in cases when the source code of some software is not available or there is not enough knowledge on all details of software implementation.

The idea of the proposed runtime verification is close enough to the formal class of runtime verification discussed in [5].

B. Data Object-Driven Data Quality Model

Data object-driven data quality model consists of 3 main components: (1) data object that defines the data which quality must be analysed, (2) data quality specification that defines conditions which must be met to admit data as qualitative, and (3) quality evaluation process that defines the procedure that must be performed to evaluate data quality [6].

Every component of the quality model is represented by flowchart-based diagrams that are easy to read, create and edit. This approach is based on three domain specific languages (DSLs) created for every model's component.

As it follows from the listed components, used solution doesn't use the concept of "dimension". Instead, the wider concept of "data quality specification" is used. The main idea of this model is that all components are fully defined by user in correspondence with users' viewpoint on the specific dataset and quality.

Data object is defined in accordance with data needed to be analysed, the parameters that do not make sense for particular users and use-cases are ignored. Data objects of the same structure form data object class where each individual data object may contain parameter values fully or partially [4]. Similarly, data quality specification is defined by user depending on the use-case. The nature of quality requirement or condition depends on the users' need. It can be compared with rule-based approaches used for relational database analysis. However, this approach reduces this limitation and can be applied to wider range of data structures. Currently, it can be applied to structured and semi-structured data. Data quality specification can be defined informally or in formal way, however at the last step all requirements are replaced by executable artefacts such as SQL statements or program code that further are executed.

Such approach is quite simple as it is very intuitive and close enough to "data" and "data quality" concepts nature. As a result, it is expected to be well-understood even by non-IT and non-data quality experts. It is one of the main benefits of this approach as usually approaches for data quality evaluation are suitable mostly for IT- and DQ- experts requiring deep knowledges in both areas [6]. However, data object-driven model can be used by wide audience without need to make in-depth analysis of the basics of the approach as it usually happens with other approaches where at least an exploration of the list of dimensions, their meanings and criteria under each of them, needs a lot of time for every particular solution as criteria differentiate from case to case [3], [6], [7]. At the same time, the proposed approach already demonstrated its effectiveness by applying it to real datasets [4], [6] – [8].

As a result, the given research uses data object-driven data quality model as the most appropriate option. In order to explain basics of the proposed solution, the next chapter summarizes the basic concepts involved in it. The required modifications are outlined to achieve desirable result.

III. A Proposed Solution

The main idea of the proposed solution is to allow check the quality of data while business process is executed after each data object update. In other words, when the process X_a step S_n is done, check data quality requirement $DQS_1(X_a, S_n)$. Data quality requirements in scope of this research are "external constraints" [9] defined for the particular process.

A. Concept of Runtime Data Quality Control

Following runtime verification mechanisms proposed by [2], a business process description should be defined for process verification. Process description contains process states and events linking states (Fig. 1). From the data quality perspective each of processes may affect one or more data objects. Accordingly, data object class definition should be added to the process definition (described in the previous section). These objects are to be changed during business process execution and there could be different verification rules for each of process steps.

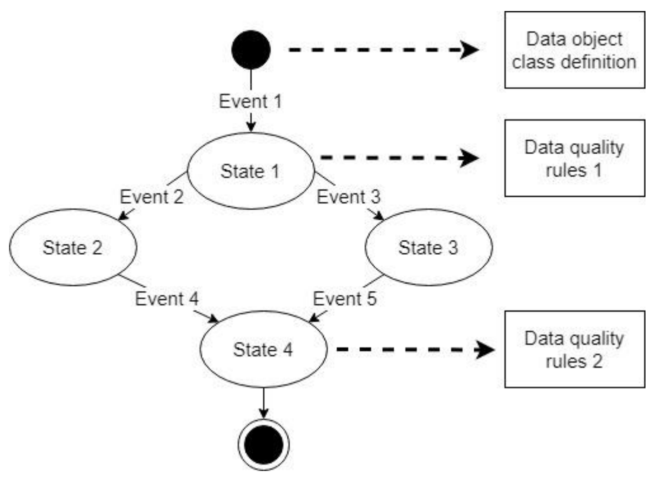

Fig. 1 Business process verification procedure

When process verification is running, new process verification instance is created by each business process start event. If necessary, corresponding data object is extracted verified according data quality definition. When next process execution event is detected, process verification instance is moved to the next state and next data object version is extracted, and its quality is verified. Thereby each process verification instance may have more than one data object instance and data quality of one data object may be verified not just at a fixed moment of time, but between its modifications accordingly (Fig. 1). This allows to identify (a) data quality loss exactly when it happens and (b) the incorrectly working process events.

According to [2] the business process verification involves two components: a *controller* and *agents*. Data quality verification adds *Data link* component to the solution (Fig. 2). Data link provides required connection to the database with business process objects and extracts data object copy when it is required by runtime verification controller. Therefore, not only data object class definition is required for data quality runtime verification, but also a definition of data object mapping to business process database: data link uses this definition for data object extraction from business process database.

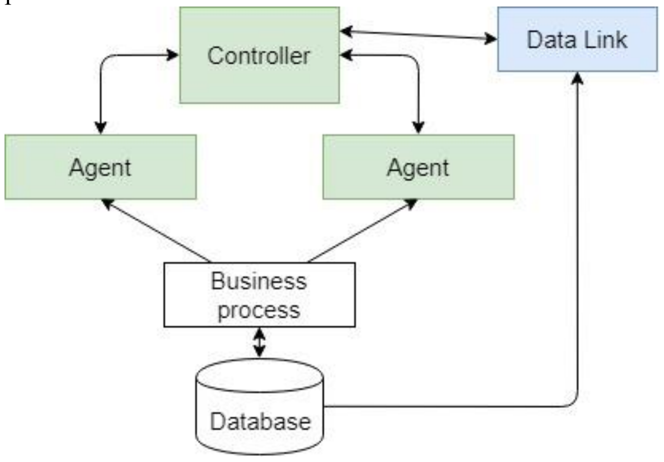

Fig. 2 Architecture of the proposed solution

B. Data Object Definition

DSL for defining of data objects is discussed in detail in [4] and [6]. As more and more customers are using electric scooter renting services, this will serve as an example to explain the proposed solution. One scooter rental case will be a data object sample. It contains data fields:

- rental ID;
- scooter code (*deviceCode*) – reference to the list of scooters available for the region;
- status (*status*) – rental status that may have one of three values: *riding*, *pause*, *finished*. When scooter is used, it always has status "*riding*". If the customer decides to stop, leave scooter on the street, and lock it for further use after some minutes, scooter is in status "*pause*". These "*pause*" minutes should be counted because another tariff may be applied for this period;
- start time (*startTime*) – time when the scooter's rental is started;
- start location (*startLocation*) – location where the rental is started;
- finish time (*finishTime*) – time when the rental is finished;
- final location (*finishLocation*) – final location of the scooter;
- pause minutes (*pauseMinutes*) – minutes spent for pauses;
- total distance (*totalDistance*) – total ride distance.

C. Data Quality Runtime Verification Process

Data quality runtime verification requires a business process definition, including states of process, possible events, and a data quality definition. The definition of the verification process for an example of the scooter rental process is shown in Fig. 3.

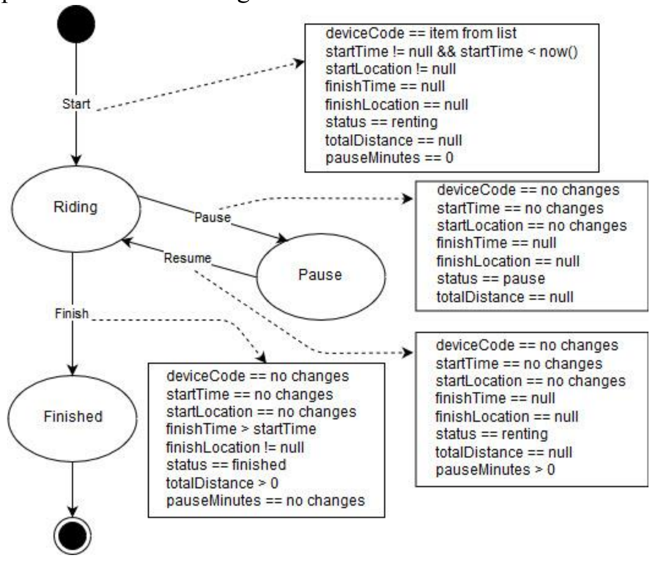

Fig. 3 Data quality verification of scooter rental process

When scooter rental starts, the initial data object should be extracted. According validation rules, the initial rental object contains information about rented device (reference to existing item from device list). *StartTime* should be different from null and less than *now()*, *startLocation* should be provided and the rental state should be "*renting*".

After pause event is detected by runtime verification controller, a new data object version for verification should be extracted from database. New rules are applied to the object:

- *deviceCode*, *startTime* and *startLocation* remain unchanged (these values are set once when the object is created);
- *status* = "*pause*";
- *pauseMinutes* should be unchanged comparing with the previous object version.

When riding activity is resumed (i.e., event "*resume*" is detected by the verification controller), the next copy of rental data object should be extracted, and a new set of rules should be applied:

- *deviceCode*, *startTime* and *startLocation* are unchanged;
- *status* = "*riding*";
- *pauseMinutes* should be more than in the last version of object;
- *finishTime* and *finishLocation* are still null.

After ride finish event is executed and detected by runtime verification controller, the last version of rental data object is extracted from the business process database. The 4th set of data quality rules should be verified:

- *deviceCode*, *startTime* and *startLocation* are unchanged;
- *status* = "*finished*";
- *pauseMinutes* are unchanged from the previous object version;
- *finishTime* and *finishLocation* are not null as the ride is finished and, moreover, *finishTime* > *startTime*.

As it can be noticed from the example, the runtime verification provides new possibilities for data quality verification:

- data quality may be applied and verified during business process execution and just for one object, not for the whole database;
- quality of data changes is verified by comparison of different versions of the same object;
- in the case of defect, the location of defect's origin may be identified.

IV. Analysis of the Proposed Solution

The presented idea allows not only to ensure the process was correct and any logical or "external" constraint weren't complied, but also to identify the moment and activity that caused or led to incorrect or inconsistent result.

As an example, let us imagine we have a database which quality we use to check once a month. We have already checked the quality of data of this database a month ago and it was of an excellent quality without any data quality issues or even anomalies. Now, a month later, we check it once again and find data quality problems not only in new records but also in those which were of good quality a month ago. It is difficult to detect the moment, when the data was changed, i.e. the qualitative data was replaced by data of poor quality, especially, if we don't have access to log files where all activities are fixed. Moreover, in some cases such log files don't exist at all or they are not detailed enough. However, the proposed idea of runtime verification in combination with data quality checks would solve this problem, detecting the moment and activity that caused the problem. To sum up, the main advantages of the proposed idea:

- data quality is verified immediately after the data is created/ modified;
- it is possible to detect the process step where the data is damaged;
- data quality evaluation is performed for the entire data set, not only for a specific data object that was changed;
- evaluation of the total data quality is reliable all the time;
- data verification can be performed independently of the system being executed.

However, there are also some potential disadvantages:

- if the runtime verification is performed incorrectly, it can lead to a tangible overload of the process being verified;
- by performing data runtime verification in parallel with system execution, data errors can be obtained for correct data if the process performs faster than the verification process and the data changes do not correspond to the step which data modifications are checked.

The proposed approach differs significantly from the Object Constraint Language (OCL) approach, which is designed to protect the database against incorrect value input but does not detect errors in input data.

As for the proposed solution, the main scope of data quality checks is data object retrieval. Data for their further quality checking are retrieved from agents by using denormalization as it significantly speeds up data retrieval. The denormalization must be implemented dynamically without knowing the denormalized relational target structure in advance [10].

The quality checking analysis runs in accordance with the initial runtime verification mechanism. In scope of the one separate check for the business process to be analysed, it can be compared with the model checking and testing mechanism using pre- and post- conditions proposed in [11]. By precondition is meant the result of previous check that is used as an input, however postcondition checking is performed when an execution of verification completes. It is obvious that preconditions should be correct to be suitable for usage in quality checks. In our case, all preconditions are considered as correct as they are analysed at the previous stages/ steps. However, another assumption is that the initial data at the initial state of a check (let call it q_0), that also is a precondition, is also correct as any data modification was done previously, as it is assumed that statically stored data (that isn't involved in any process) is checked from time to time and as a result is correct.

The frequency and number of data quality checks as well as points when they should be done depend on the use-case and user's preferences. This idea corresponds with the data object-driven approach to data quality evaluation allowing users to take control over every step of data quality analysis process. As a result, the proposed solution respects quality checks: (a) after every step as well as (b) only when the user considers them as important, or (c) with periodical frequency after a particular step, for instance, once a day or every time after specific process is finished etc. The first option ensures in-depth and comprehensive quality analysis, when every step is checked. However, as there might be cases, when several steps are not of high importance at least for a particular user, or there is no necessity in continuous checks, for instance, in order to save resources and efficiency, the second and third options appear suitable. Moreover, in the future, the idea of prioritization mechanism would be evaluated to provide users the possibility to perform some checks with higher priority first. This mechanism would offer to users a higher level of control over the whole process.

The number of cases when the proposed solution can be useful is high, including the continuous assessment of the quality of information systems, e-government [12], etc. by data quality runtime verification. The proposed solution can lead to the improvement of quality of many services increasing government effectiveness and quality of public services [13] that nowadays become a topical issue.

V. CONCLUSION

This paper deals with runtime verification for checking data quality during business process execution. The user defines data objects and requirements of his/ her interest using graphic DSLs and ensuring high quality of data object parameter values.

Unlike other data quality studies, the proposed solution provides an operational runtime verification of data quality requirements, allowing to detect deviations from quality requirements in the particular business process's execution step when incorrect data object parameters are recorded in the database. Verification of data quality requirements and the base process are parallel processes. Impact of verification on the base process performance is insignificant and acceptable if sufficient hardware capacity is available.

The proposed solution is an "external" solution that checks the data quality requirements without direct connection to the business process. Such approach allows enabling and disabling of runtime verification of data quality requirements operationally at various stages of information system usage. Graphical DSLs that is used to describe data objects and quality requirements is intuitively understood and suited for use by non-IT and data quality specialists.

In the future, the proposed approach might be applied to issues of the semantic web.

REFERENCES

[1] El Hadji Bassirou Toure, I. Fall, A. Bah, M. S. Camara, *Megamodel-based Management of Dynamic Tool Integration in Complex Software Systems.* In FedCSIS Position Papers, 2016, pp. 211-218, http://dx.doi.org/10.15439/2016F585.

[2] I. Oditis, J. Bicevskis, *Asynchronous Runtime Verification of Business Processes: Proof of Concept*. International Journal of Simulation-Systems, Science & Technology, 2015, 16(6), 1-11, http://dx.doi.org/10.5013/IJSSST.a.16.06.06.

[3] J. Bicevskis, Z. Bicevska, A. Nikiforova, I. Oditis, *An Approach to Data Quality Evaluation*. In 2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS), IEEE, 2018, pp. 196-201, http://dx.doi.org/10.1109/SNAMS.2018.8554915.

[4] J. Bicevskis, Z. Bicevska, A. Nikiforova, I. Oditis, *Data quality evaluation: a comparative analysis of company registers' open data in four European countries*. In Communication Papers of the Federated Conference on Computer Science and Information Systems (FedCSIS), 2018, pp. 197-204, http://dx.doi.org/10.15439/2018F92.

[5] A. Coronato, A. Testa, *Approaches of Wireless sensor network dependability assessment.* In 2013 Federated Conference on Computer Science and Information Systems, IEEE, 2013, pp. 881-888.

[6] A. Nikiforova, *Open Data Quality Evaluation: A Comparative Analysis of Open Data in Latvia*. Baltic Journal of Modern Computing, 2018, 6(4), 363-386, https://doi.org/10.22364/bjmc.2018.6.4.04.

[7] A. Nikiforova, J. Bicevskis, *An Extended Data Object-driven Approach to Data Quality Evaluation: Contextual Data Quality Analysis*. In Proceedings of the 21st International Conference on Enterprise Information Systems - Volume 1: ICEIS, 274-281, 2019, http://dx.doi.org/10.5220/0007838602740281.

[8] A. Nikiforova, *Analysis of Open Health Data Quality Using Data Object-Driven Approach to Data Quality Evaluation: Insights from a Latvian Context.* In IADIS International Conference e-Health 2019, Part of the IADIS Multi Conference on Computer Science and Information Systems, MCCSIS 2019, IADIS

[9] G. C. Deka, *NoSQL: database for storage and retrieval of data in cloud*, Ed. CRC Press, 2017, https://doi.org/10.1201/9781315155579.

[10] C. Gröger, F. Niedermann, B. Mitschang. *Data mining-driven manufacturing process optimization*. In Proceedings of the world congress on engineering, 2012, Vol. 3, pp. 4-6.

[11] S. Khurshid, C. S. Păsăreanu, W. Visser, *Generalized symbolic execution for model checking and testing*. In International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, Berlin, Heidelberg, 2003, pp. 553-568, https://doi.org/10.1007/3-540-36577-X_40.

[12] E. Ziemba, T. Papaj, D. Descours, *Assessing the quality of e-government portals-the Polish experience.* In 2014 Federated Conference on Computer Science and Information Systems, IEEE, 2014, pp. 1259-1267, http://dx.doi.org/10.15439/2014F121.

[13] A. Karabegovic, M. Ponjavic, *Geoportal as decision support system with spatial data warehouse.* In 2012 Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, 2012, pp. 915-918, http://dx.doi.org/10.13140/RG.2.2.26385.68963.